CPSC 319 Assignment 5 Questions Ethan Reed - 30123128

Q1: Briefly describe your hash function and how it works.

A1: The hash function traverses through the characters in the input string and multiplies the character’s integer Unicode value by 3i where i is the index of the character. After calculating a value for each character, the values are added together to form the hash. The function returns **hash *mod* arr.length**, so that the resulting index is not out of bounds. The reason that the Unicode value of each character is multiplied by ni, is to give weight to each character’s position in the string, 3 was chosen for n through experimentation. In order to save some time, the hash function will only look at the first 10 characters in any longer strings, the value 10 was chosen through experimentation.

Q2: Analyze the quality of your hash function based on the statistics reported. How could

you improve it?

A2: The hashing efficiency was around 64% with the provided input file, that means that more than 1 in every 3 inserts resulted in a collision. The function could likely be improved through further experimentation with the value of n, or by making n some function of i or each characters Unicode value. The number of characters read could also be increased, so that strings such as “razzmatazz” and “razzmatazzle” would not result in the same hash. The hashing efficiency can also be increased by modifying parameters outside of the hash function, such as by decreasing the load threshold at which the table is resized. Collisions can also be reduced by improving collision handling, as seen in the bonus activity.

Q3: Compare your hash function with an ideal hash function. How close is it to ideal performance?

A3: An ideal hash function would never result in collisions, as stated in A2, my hash function results in collisions for around 33% of all inserts. My function is not all that close to ideal performance.

Q4: If the input data set contained numerals (for example, student ID numbers) instead of

alphabetical characters, how would it influence the performance of your hash

function? Justify your answer with a brief analysis.

A4: The hash function would still work, as each digit has an integer Unicode value attributed to it, and the performance should be mostly unchanged. Although there are only 10 unique numerical characters (compared to 52 unique alphabetical characters), my hash table does not allow for duplicates. Each input to the hash function will still be unique, and there will be minimal overlap between hash values.

BONUS: The hashing efficiency was barely improved by using quadratic probing (63.6% vs 64.0%), but each collision resulted in a much shorter chain of reads. The longest read chain when linear probing was more than 100 times the length of the longest chain using quadratic probing (3103 vs 26). On average, each insert required around 3 reads when using quadratic probing, this is much less than the around 112 reads that the average insert would require when using linear probing. Overall, the performance was vastly improved by switching from linear probing to quadratic probing.